

Post-Mortem

Assignment 04

February 19, 2019

We normally publish the post-mortem for an assignment after it has been marked and released. Here is a list of common errors provided by the graders for assignment 4.

General

- Students are highly encouraged to make sure their code runs before submitting it.
- Many students were missing examples/tests for base case(s) throughout the assignment. When writing a recursive function, there should be an example/test for every base case, and an example/test for at least one of the recursive cases, as discussed in the style guide.
- Types in contracts should be correctly formatted as per the style guide.
- Where applicable, contract types should be as specific as possible. For example, a function that consumes a `Rental` should be represented with `Rental` in the contract, and not `(anyof Bicycle Boat Horse)`.
- Contracts that consumed or produced lists often had formatting errors. For example, if a function consumes a list of numbers, it should be indicated as `(listof Num)` in the contract **exactly**, and not `listof(Num)`, `list of Num`, `list-num`, `(listof positive)`, `list`, or any other variation.
- This assignment did not involve the use of `Posns` in any way. Thus, a correct solution to any of the assignment problems would not include `Posns`.

Question 2

- Many students used `(anyof Bicycle Boat Horse)` instead of `Rental` in their contract. If there is a user-defined type, contracts should include it everywhere applicable.
- Many students did not define constants for the `Rental` rates or `Rental` capacities associated with a `Bicycle`, `Boat`, or `Horse`.
- Many students did not include a contract for their `rental-template`. As discussed in the style guide, templates should always include a contract. Moreover, many students contract did not reflect the fact that the result of a template function is `Any`.
- Many students had templates that did not include the selectors for the `Bicycle`, `Boat` and `Horse` structures.
- In part (c), some students did not make sure that a `Horse` rental was valid only if the consumed number of renters did not exceed the capacity of a `Horse`, and that the consumed duration did not exceed the `Horse`'s stamina.

- In part (d), some students did not correctly communicate in their requirements that the `Rental` must be valid according to the consumed parameters.
- In part (d), some students did not correctly account for the case where the number of renters is greater than 3 for a `Horse`.

Question 3

- In part (b), some students did not check the case that that a list contains only a single integer. e.g. `(cons 1 empty)`
- In part (c), some students missed the non-empty list requirement for `geometric mean`.
- Helper functions were often ambiguously named.

Question 4

- Many students were missing some (or all) of the requirements for a valid `Pie`.
- In particular, many students forgot to mention that `size is` (`anyof 'small 'medium 'large`).
- Many students copied the text on the assignment word-for-word instead of formatting it accordingly to be more in line with the style guide.
- In part (b), many students wrote `Pie ->Bool` as their contract, when it should be `Any ->Bool`.
- In part (b), many students forgot to check if all the fillings are symbols.
- In part (b), many students forgot to check for all the requirements on the fillings that make a pie valid.
- In part (c), many students forgot that `false` is also a result of `pie-swap` hence the contract should be `Pie ->(anyof Pie false)`.
- In part (c), many students did not check to see if the result of replacing one filling with another produces a valid `Pie`.

Ongoing Errors

The following is a list of common errors from previous assignments that were still repeated for assignment 4.

- Many students are still missing parameter references in their purpose statements. Purpose statements should meaningfully use each parameter name, and the parameter names should be written **exactly** as they appear in the function header.
- Constants (and helper functions) should be defined **above** the design recipe for the function they are used in. Some students are still defining their constants and helper functions between their examples and function definition, or after the main functions specified in the assignment.
- All design recipe components, except for tests, are required for helper functions. Some students did not include these design recipe components for their helper functions.
- Starting a new `cond` expression immediately in an `else` clause is unneeded. Instead, directly check for the next condition in the original `cond` expression.